

INTRODUCCIÓN



paco@portadaalta.es

Índice

- **Entorno de desarrollo**
- **Estructura de un proyecto Android**
- **Componentes de una aplicación Android**
- **Ejemplos:**
 - ✓ **Mi primera aplicación**
 - ✓ **Divisas**
 - ✓ **2 actividades**
 - ✓ **Contador de cafés**

Entorno de desarrollo

Vamos a describir los pasos básicos para disponer en nuestro PC del entorno y las herramientas necesarias para comenzar a programar aplicaciones para la plataforma Android.

Paso 1. Descarga e instalación de Java.

Si aún no tienes instalado ninguna versión del JDK (Java Development Kit) puedes descargar la última versión desde la web de [Oracle](https://www.oracle.com/technetwork/java/javase-downloads-1344942.html).

Paso 2. Descarga e instalación de Eclipse.

Si aún no tienes instalado Eclipse, puedes descargar la última versión desde [eclipse.org](https://www.eclipse.org/). Conviene descargar la versión Eclipse IDE for Java Developers.



Eclipse IDE for Java Developers, 154 MB

Downloaded 604,311 Times

The essential tools for any Java developer, including a Java IDE, a CVS client, Git client, XML Editor, Mylyn, Maven integration...



Linux 32 Bit
Linux 64 Bit

Entorno de desarrollo

Paso 3. Descargar el SDK de Android.

El SDK de la plataforma Android se puede descargar desde developer.android.com

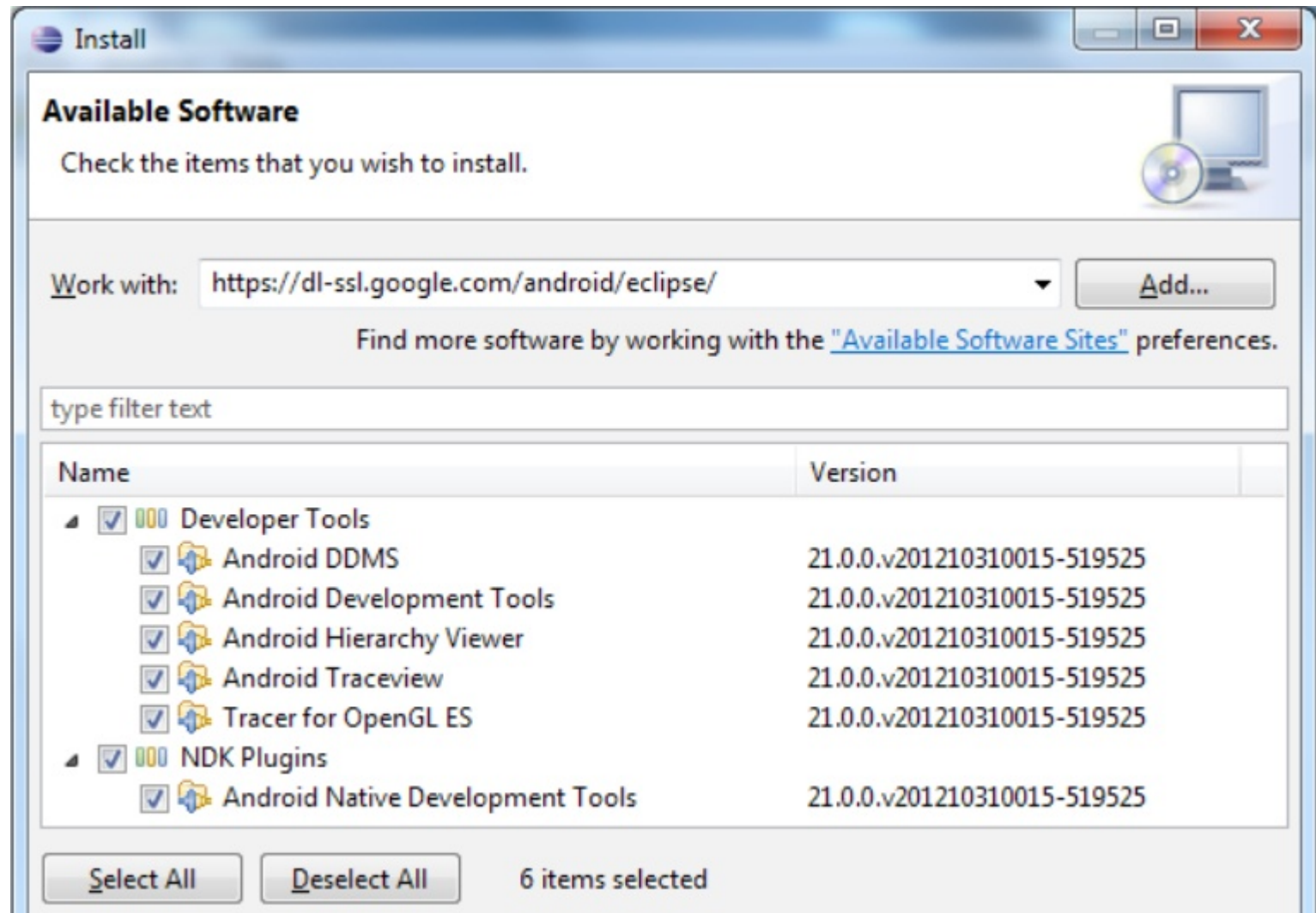
Paso 4. Descargar el plugin de Android para Eclipse.

Google pone a disposición de los desarrolladores un plugin para Eclipse llamado Android Development Tools (ADT) que facilita en gran medida el desarrollo de aplicaciones para la plataforma.

Se puede descargar mediante las opciones de actualización de Eclipse, accediendo al menú "Help / Install new software..." e indicando la siguiente URL de descarga:

<https://dl-ssl.google.com/android/eclipse/>

Entorno de desarrollo

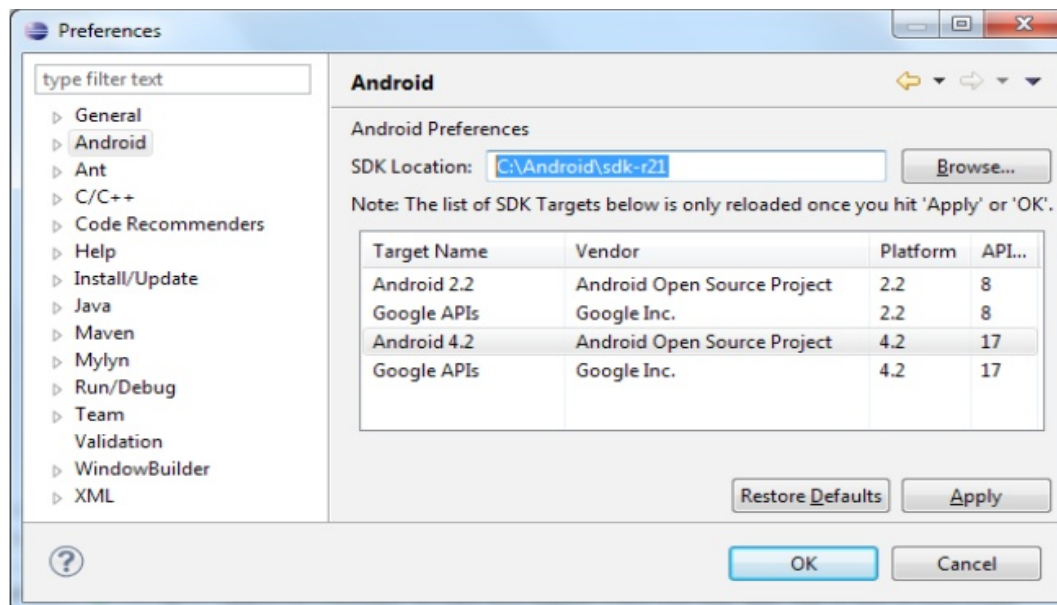


Entorno de desarrollo

Paso 5. Configurar el plugin ADT.

Una vez instalado el plugin, tendremos que configurarlo indicando la ruta en la que hemos instalado el SDK de Android. Para ello, iremos a la ventana de configuración de Eclipse (Window / Preferences...), y en la sección de Android indicaremos la ruta en la que se ha instalado.

Finalmente pulsaremos OK para aceptar los cambios. Si aparece algún mensaje de warning aceptamos sin más, ya que se son problemas que se solucionarán en el siguiente paso.



Entorno de desarrollo

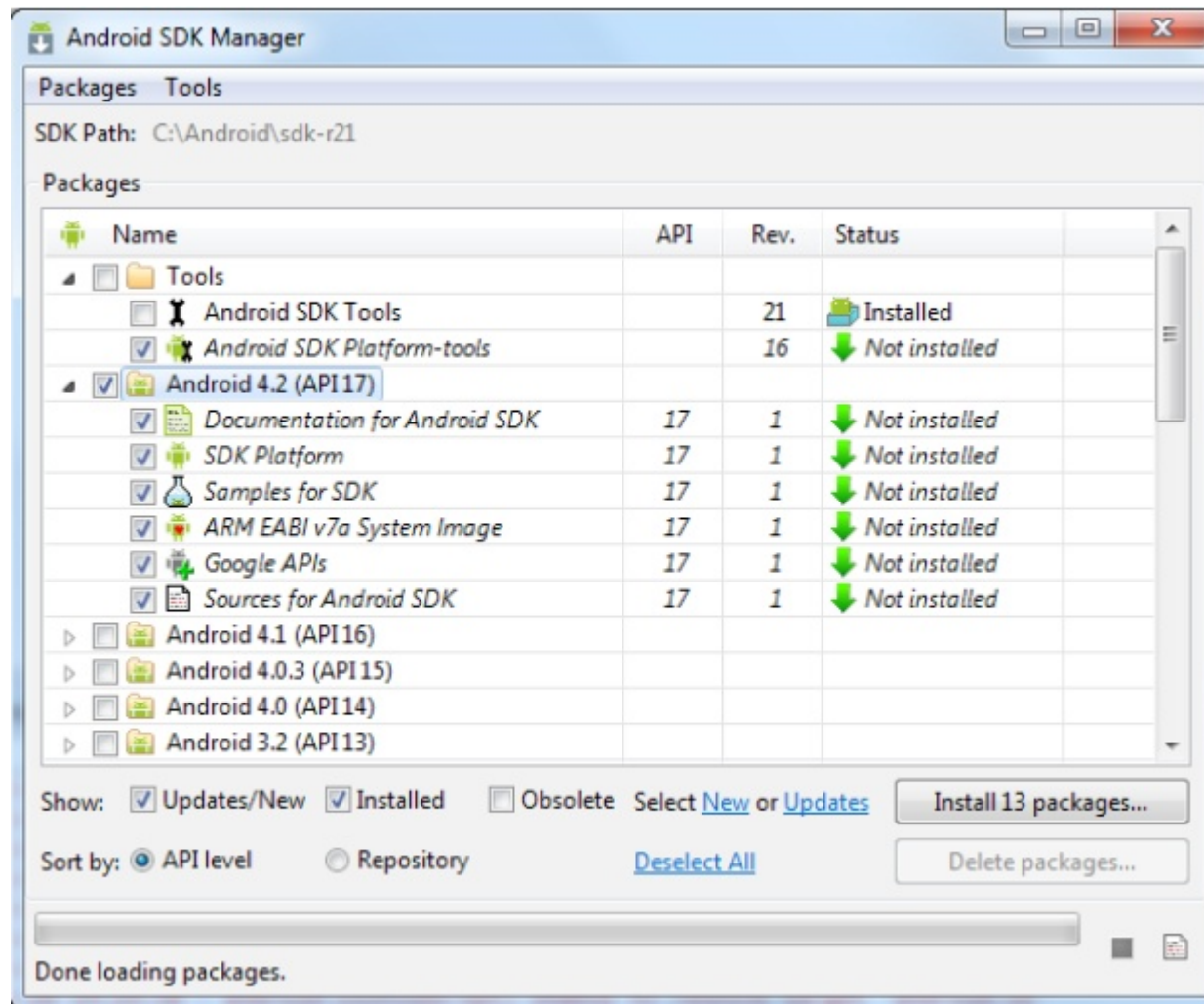
Paso 6. Instalar las Platform Tools y los SDK Platforms necesarios.

Además del SDK de Android comentado en el paso 2, que contiene las herramientas básicas para desarrollar en Android, también deberemos descargar las llamadas Platform Tools, que contiene herramientas específicas de la última versión de la plataforma, y una o varias plataformas (SDK Platforms) de Android, que no son más que las librerías necesarias para desarrollar sobre cada una de las versiones concretas de Android.

Así, si queremos desarrollar por ejemplo para Android 2.2 tendremos que descargar su plataforma correspondiente. Es conveniente instalar al menos 2 plataformas: la correspondiente a la última versión disponible de Android, y la correspondiente a la mínima versión de Android que queremos que soporte nuestra aplicación.

Para ello, desde Eclipse debemos acceder al menú "*Window / Android SDK Manager*". En la lista de paquetes disponibles seleccionaremos las "*Android SDK Platform-tools*", las plataformas "*Android 4.2 (API 17)*" y "*Android 2.3 (API 10)*", y el paquete extra "*Android Support Library*", que es una librería que nos permitirá utilizar en versiones antiguas de Android características introducidas por versiones más recientes. Pulsaremos el botón "*Install packages ...*" y esperaremos a que finalice la descarga.

Entorno de desarrollo



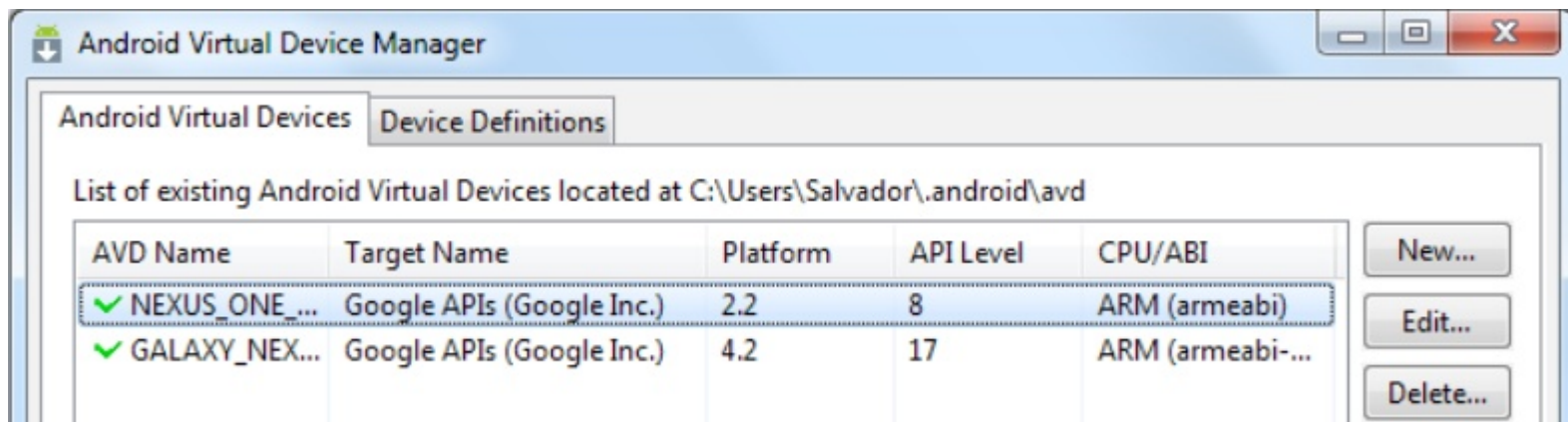
Entorno de desarrollo

Paso 7. Configurar un AVD.

A la hora de probar y depurar aplicaciones Android no tendremos que hacerlo necesariamente sobre un dispositivo físico, sino que podremos configurar un emulador o dispositivo virtual (Android Virtual Device, o AVD) donde poder realizar fácilmente estas tareas.

Para ello, accederemos al AVD Manager (menú Window / AVD Manager), y en la sección Virtual Devices podremos añadir tantos AVD como se necesiten (por ejemplo, configurados para distintas versiones de Android o distintos tipos de dispositivo).

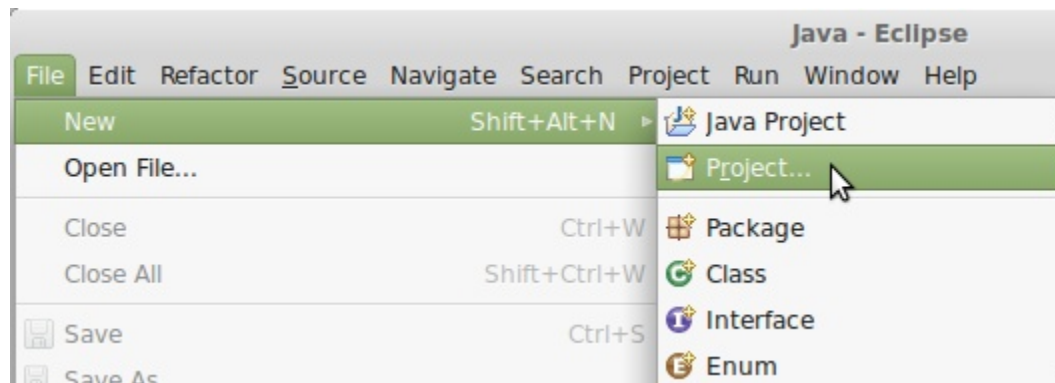
Conviene configurar al menos dos AVD, uno para la mínima versión de Android que queramos soportar, y otro para la versión más reciente disponible.



Estructura de un proyecto Android

Para empezar a comprender cómo se construye una aplicación Android vamos a crear un nuevo proyecto Android en Eclipse y echaremos un vistazo a la estructura general del proyecto creado por defecto.

Para crear un nuevo proyecto abriremos Eclipse e iremos al menú File / New / Project.



Estructura de un proyecto Android

De esta forma iniciaremos el asistente de creación del proyecto, que nos guiará por las distintas opciones de creación y configuración de un nuevo proyecto.

En la primera pantalla indicaremos el nombre de la aplicación, el nombre del proyecto y el paquete java que utilizaremos en nuestras clases java.



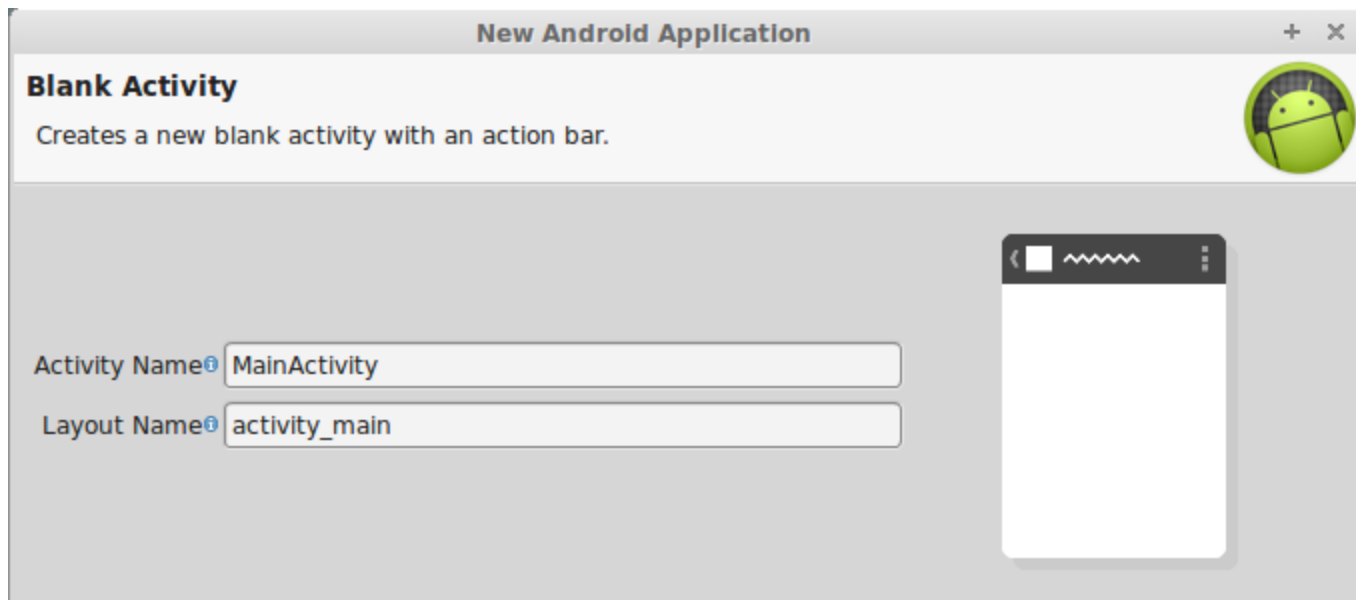
The screenshot shows the 'New Android Application' dialog box. It has a title bar with the text 'New Android Application' and standard window controls. Below the title bar, there is a subtitle 'New Android Application' and a warning icon with the text 'The application name for most apps begins with an uppercase letter'. To the right of the subtitle is the Android robot icon. The dialog contains several input fields and dropdown menus:

- Application Name:** A text field containing 'hola'.
- Project Name:** A text field containing 'hola'.
- Package Name:** A text field containing 'com.example.hola'.
- Minimum Required SDK:** A dropdown menu showing 'API 10: Android 2.3.3 (Gingerbread)'.
- Target SDK:** A dropdown menu showing 'API 20: Android 4.4 (KitKat Wear)'.
- Compile With:** A dropdown menu showing 'API 20: Android 4.4 (KitKat Wear)'.
- Theme:** A dropdown menu showing 'Holo Light'.

Estructura de un proyecto Android

En las siguientes pantallas del asistente configuraremos las diferentes opciones del proyecto (el icono, la creación de la actividad principal, etc).

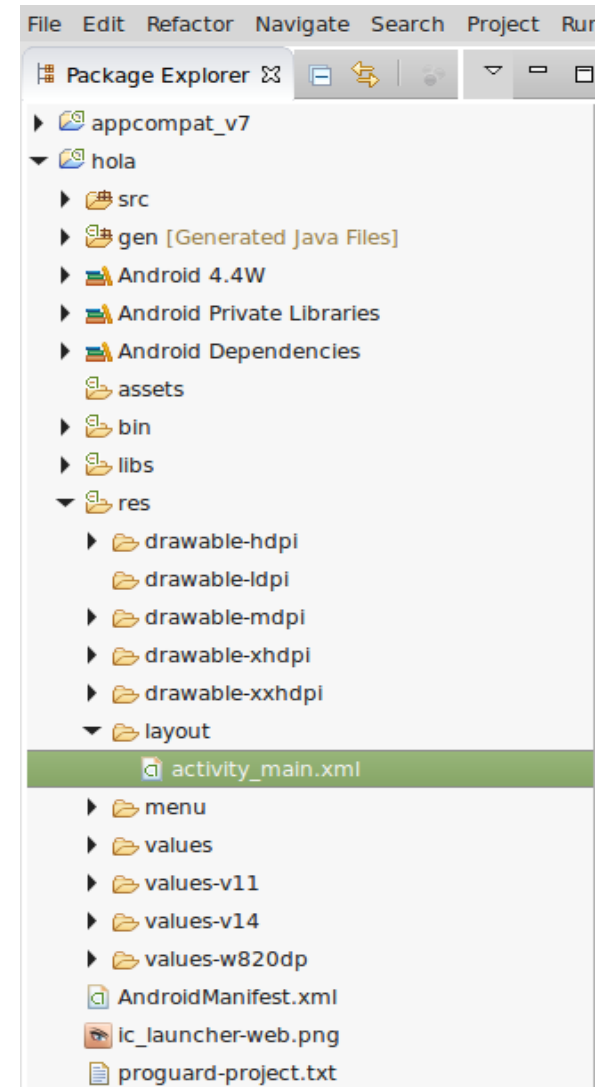
Por último, en el último paso del asistente indicaremos los datos de esta actividad principal que acabamos de elegir, indicando el nombre de su clase java asociada y el nombre de su layout (algo así como la interfaz gráfica de la actividad en formato xml).



Estructura de un proyecto Android

Una vez configurado todo, pulsamos el botón Finish y Eclipse creará por nosotros toda la estructura del proyecto y los elementos indispensables que debe contener.

En la imagen vemos los elementos creados inicialmente para un nuevo proyecto Android:



Estructura de un proyecto Android

Los elementos principales de esta estructura son:

Carpeta /src

Esta carpeta contendrá todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc. Inicialmente, Eclipse creará por nosotros el código básico de la pantalla (Activity) principal de la aplicación, que recordemos que en nuestro caso era MainActivity, y siempre bajo la estructura del paquete java definido.

Carpeta /res

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc.

Se puede consultarse la lista completa en la [documentación oficial del Android](#)

Fichero AndroidManifest.xml

Contiene la definición en XML de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, icono, ...), sus componentes (pantallas, mensajes, ...), las librerías auxiliares utilizadas o los permisos necesarios para su ejecución.

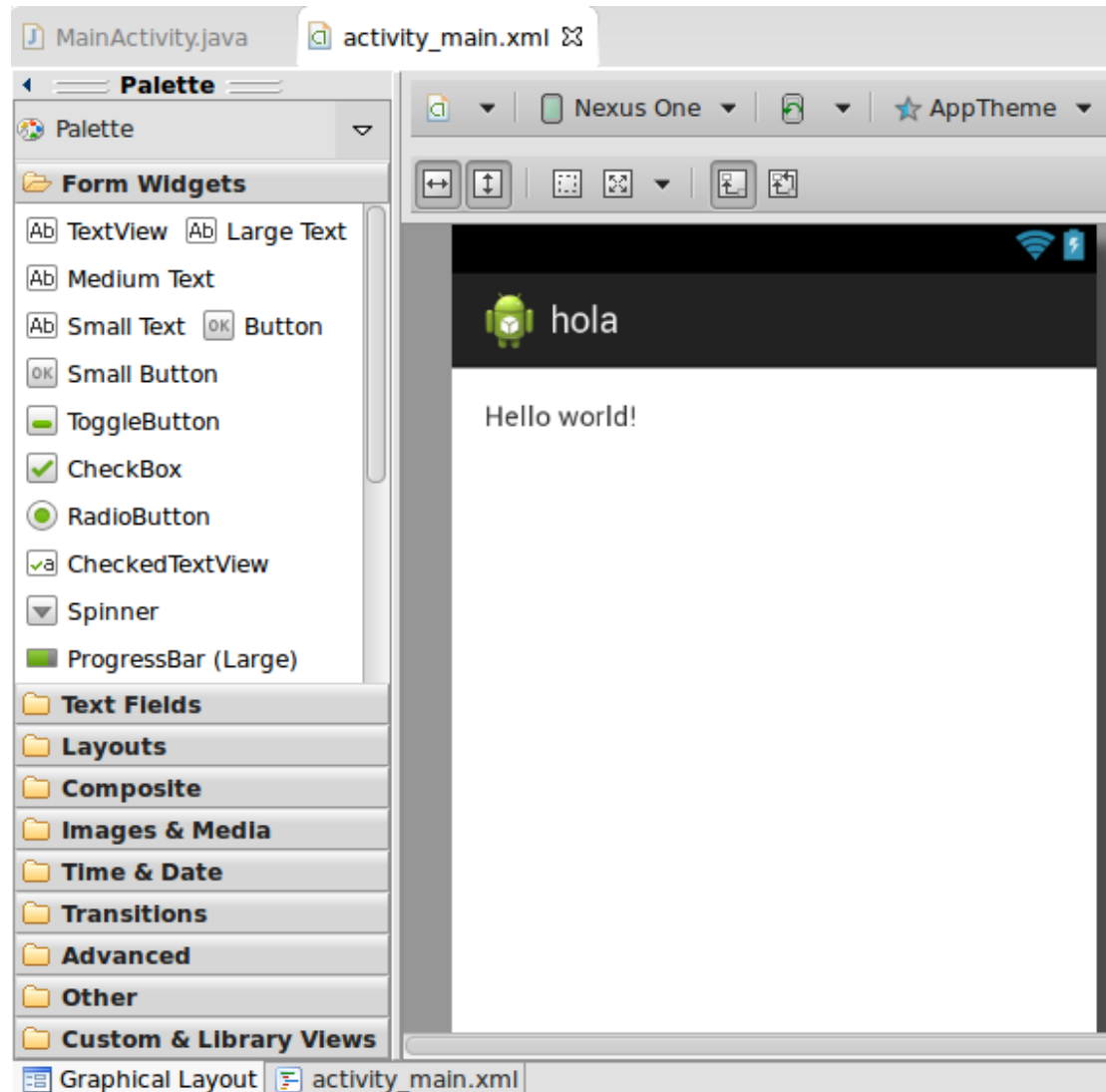
Estructura de un proyecto Android

Entre los recursos creados por defecto, cabe destacar el layout "activity_main.xml", que contiene la definición de la interfaz gráfica de la pantalla principal de la aplicación.

Si hacemos doble clic sobre el fichero Eclipse nos mostrará esta interfaz en su editor gráfico (tipo arrastrar y soltar) y como podremos comprobar, en principio contiene tan sólo una etiqueta de texto centrada en pantalla con el mensaje "Hello World!".

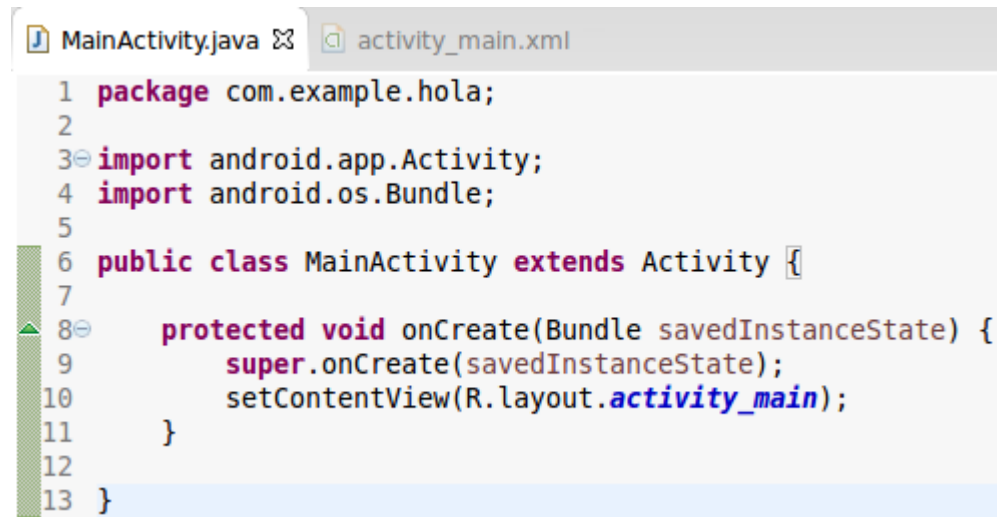
En lugar de usar el editor gráfico se puede modificar directamente el fichero XML asociado (al que se puede acceder pulsando sobre la pestaña inferior derecha, en activity_main.xml junto a la solapa "Graphical Layout").

Estructura de un proyecto Android



Estructura de un proyecto Android

Y éste es el código de nuestra primera aplicación:

A screenshot of an IDE window showing the MainActivity.java file. The window has two tabs: MainActivity.java and activity_main.xml. The MainActivity.java tab is active, displaying the following Java code:

```
1 package com.example.hola;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5
6 public class MainActivity extends Activity {
7
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12
13 }
```

Componentes de una aplicación Android

Vamos a ver los distintos tipos de componentes de software con los que podremos construir una aplicación Android.

En Java o .NET estamos acostumbrados a manejar conceptos como ventana, control, eventos o servicios como los elementos básicos en la construcción de una aplicación.

En Android vamos a disponer de esos mismos elementos básicos aunque con un pequeño cambio en la terminología y el enfoque.

Repasemos los componentes principales que pueden formar parte de una aplicación Android

Componentes de una aplicación Android

Activity

Las actividades (activities) representan el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una actividad como el elemento análogo a una ventana o pantalla en cualquier otro lenguaje visual.

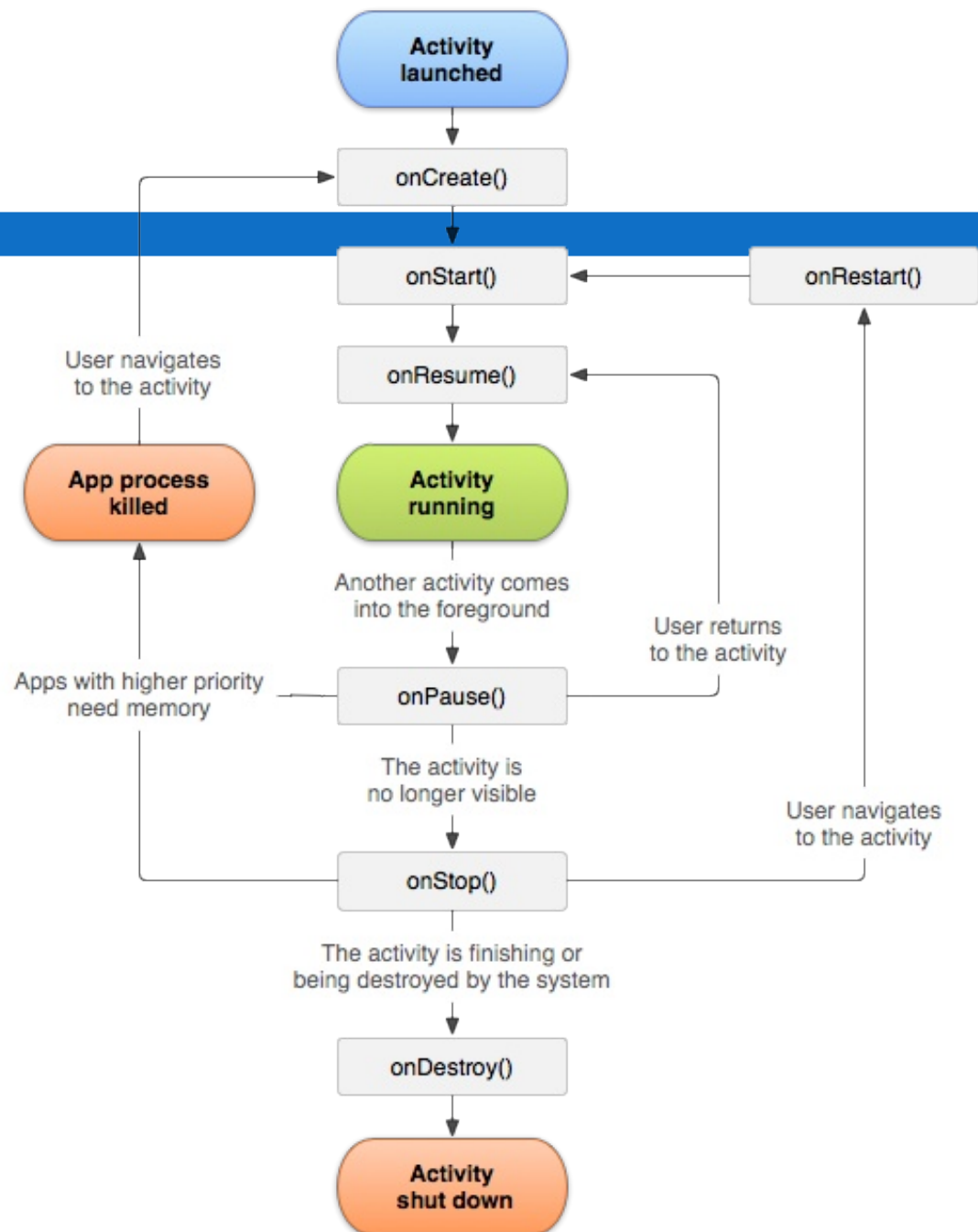
Más información en developer.android.com

Creación de una Actividad:

Para crear una actividad, se debe crear una subclase de la clase Activity(o una subclase existente). En tu subclase debes implementar los métodos a los que llama el sistema cuando la actividad realiza transiciones entre los diferentes estados de su ciclo de vida, tales como cuando la actividad es creada, parada, reiniciada o destruida.El método más importante es onCreate().

onCreate()

Se debe implementar este método. El sistema lo llama cuando se crea la actividad. En tu implementación debes inicializar los componentes principales de tu actividad. Lo más importante es llamar al método setContentView() para definir el layout para la interfaz de usuario de la actividad.



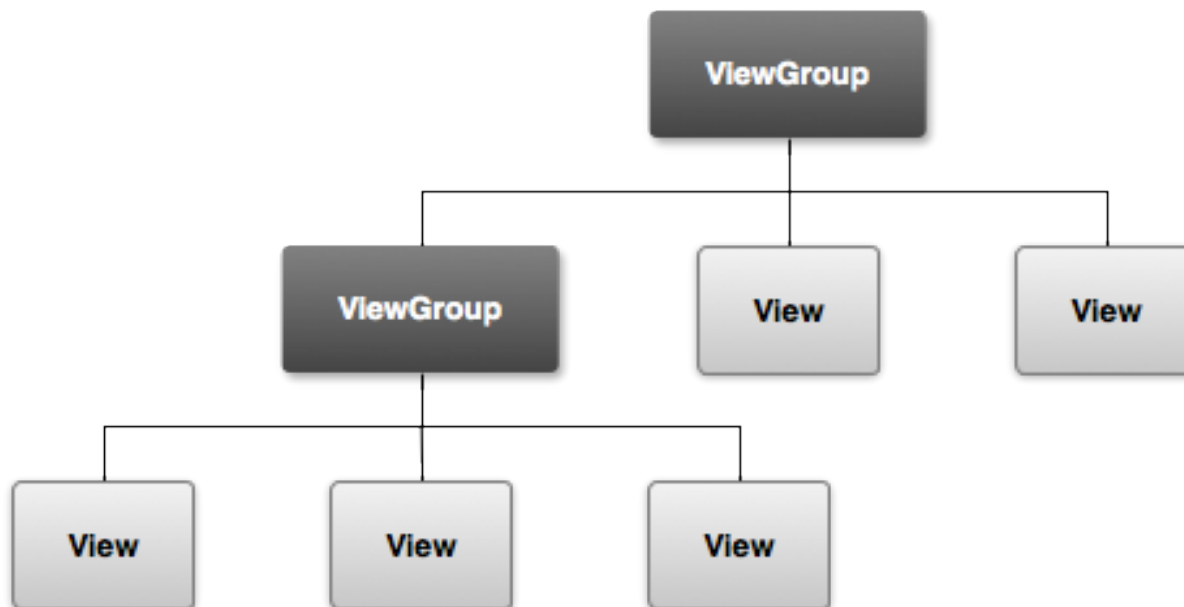
Componentes de una aplicación Android

View

Las vistas (view) son los componentes básicos con los que se construye la interfaz gráfica de la aplicación, análogo por ejemplo a los controles de Java o .NET.

De inicio, Android pone a nuestra disposición una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

Más información en [Building a Simple User Interface](#)



Componentes de una aplicación Android

Intent

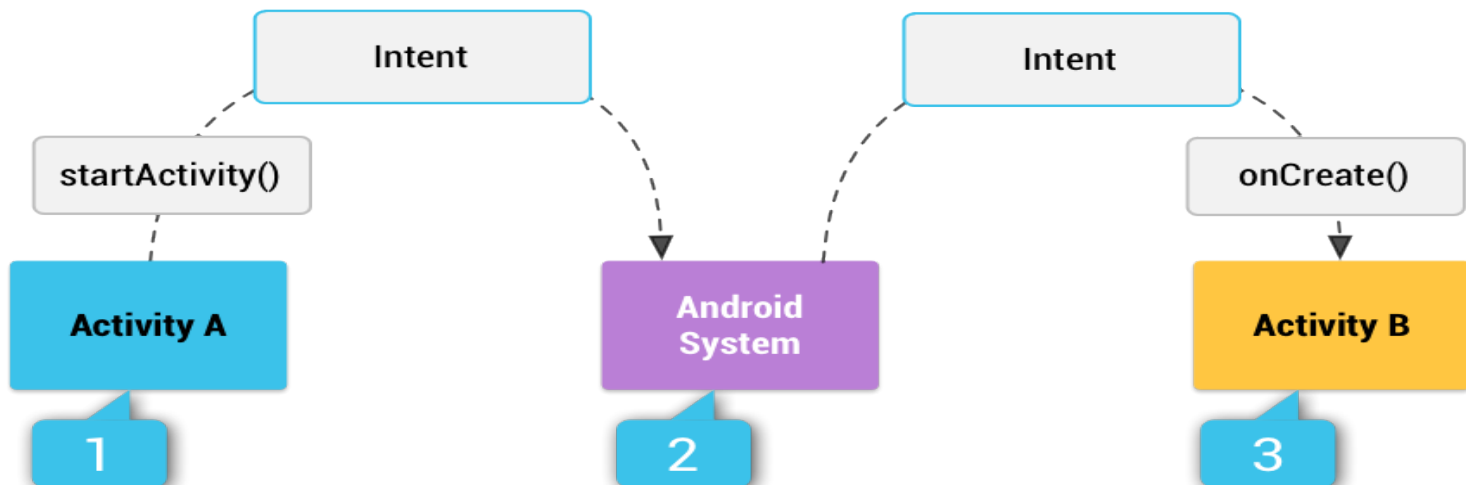
Un intent es el elemento básico de comunicación entre los distintos componentes Android. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones.

Mediante un intent se puede mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.

Hay 2 tipos de intents:

- Implícitos
- Explícitos

Más información en [Intents and Intent Filters](#)



Componentes de una aplicación Android

Service

Los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano. En concepto, son similares a los servicios presentes en cualquier otro sistema operativo. Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (p.ej. actividades) si se necesita en algún momento la interacción con el usuario.

Content Provider

Un content provider es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones. Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación. De la misma forma, nuestra aplicación podrá acceder a los datos de otra a través de los content provider que se hayan definido.

Broadcast Receiver

Un broadcast receiver es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: "Batería baja", "SMS recibido", "Tarjeta SD insertada", ...) o por otras aplicaciones (cualquier aplicación puede generar mensajes (intents, en terminología Android) broadcast, es decir, no dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlo).

Widget

Los widgets son elementos visuales, normalmente interactivos, que pueden mostrarse en la pantalla principal (home screen) del dispositivo Android y recibir actualizaciones periódicas. Permiten mostrar información de la aplicación al usuario directamente sobre la pantalla principal.

Mi primera aplicación

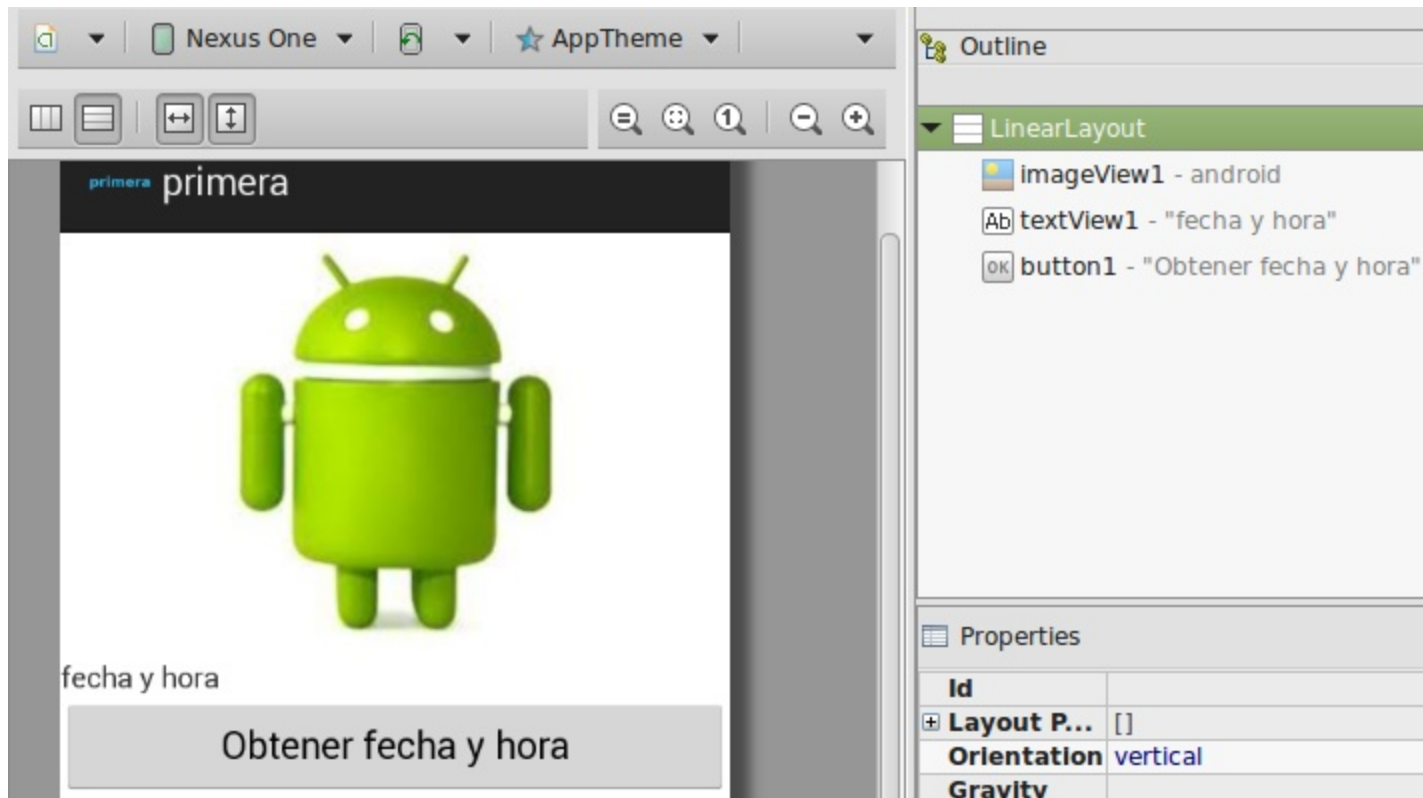
Crear una aplicación que muestre la fecha y hora actual cada vez que se pulse un botón



Mi primera aplicación

Se creará una nueva aplicación.

Se añadirán al layout una imagen (en la carpeta drawable), una etiqueta y un botón



Mi primera aplicación

El fichero xml del layout será similar a este:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >
```

```
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/android"
        android:contentDescription="@string/foto"/>
```

```
    <TextView
        android:id="@+id/textView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/fecha" />
```

```
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/obtener" />
```

```
</LinearLayout>
```

Fichero strings.xml en la carpeta values

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">primera</string>
    <string name="hello_world">Hello world!</string>
    <string name="fecha">fecha y hora</string>
    <string name="obtener">Obtener fecha y hora</string>
    <string name="foto">imagen de android</string>
</resources>
```

Mi primera aplicación

En el código se usará la interfaz `OnClickListener`, se modificará el método `onCreate` y se añadirán los métodos `onClick(View v)` y `actualizar`:

```
public class MainActivity extends Activity implements OnClickListener {  
    //Definir objetos  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        //Hacer referencia a los elementos de la vista  
    }  
  
    public void onClick(View v){  
        //Tratar el evento click del botón  
    }  
  
    private void actualizar(){  
        //Actualizar la fecha y hora  
    }  
}
```

Mi primera aplicación

```
public class MainActivity extends Activity implements OnClickListener {
    Button boton;
    TextView texto;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        boton = (Button) findViewById(R.id.button1);
        texto = (TextView) findViewById(R.id.textView1);
        boton.setOnClickListener(this);
        actualizar();
    }

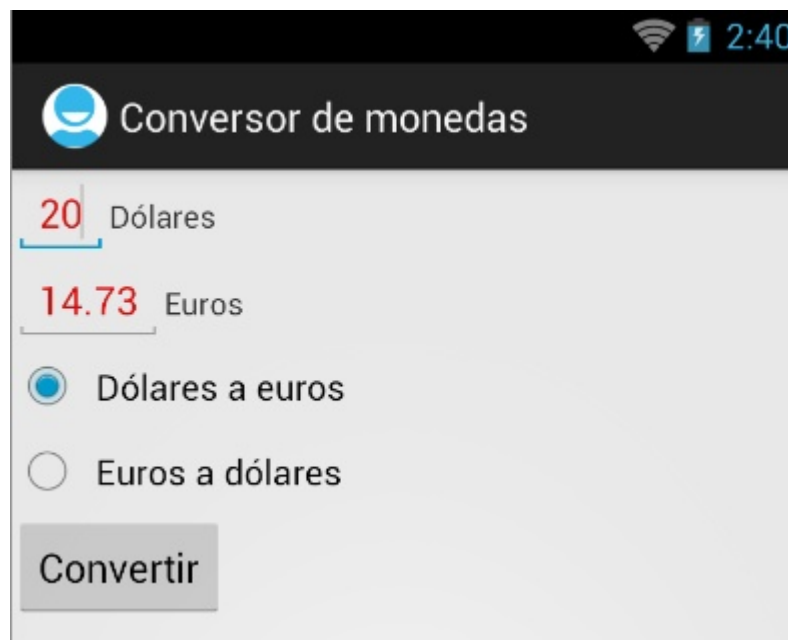
    public void onClick(View v){
        if (v == boton)
            actualizar();
    }

    private void actualizar(){
        texto.setText(new Date().toString());
    }
}
```

Divisas

Crear una aplicación que permita convertir dólares a euros y viceversa

Divisas



A screenshot of a mobile application titled "Conversor de monedas". The app interface is shown on a screen with a status bar at the top displaying a Wi-Fi icon, a battery icon, and the time 2:40. The app's header is dark grey with a blue circular icon containing a white smiley face and the text "Conversor de monedas". Below the header, there are two input fields. The first field contains the number "20" in red, followed by the text "Dólares". The second field contains the number "14.73" in red, followed by the text "Euros". Below these fields, there are two radio button options: "Dólares a euros" (which is selected with a blue dot) and "Euros a dólares". At the bottom of the app interface is a grey button with the text "Convertir".

20 Dólares

14.73 Euros

☒ Dólares a euros

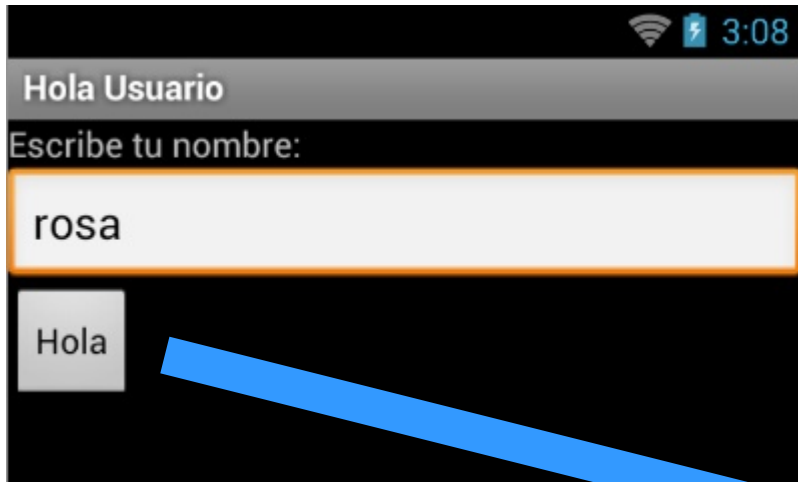
☐ Euros a dólares

Convertir

2 actividades

Crear una aplicación que pida un nombre y muestre en una segunda ventana un saludo con ese nombre

2 actividades



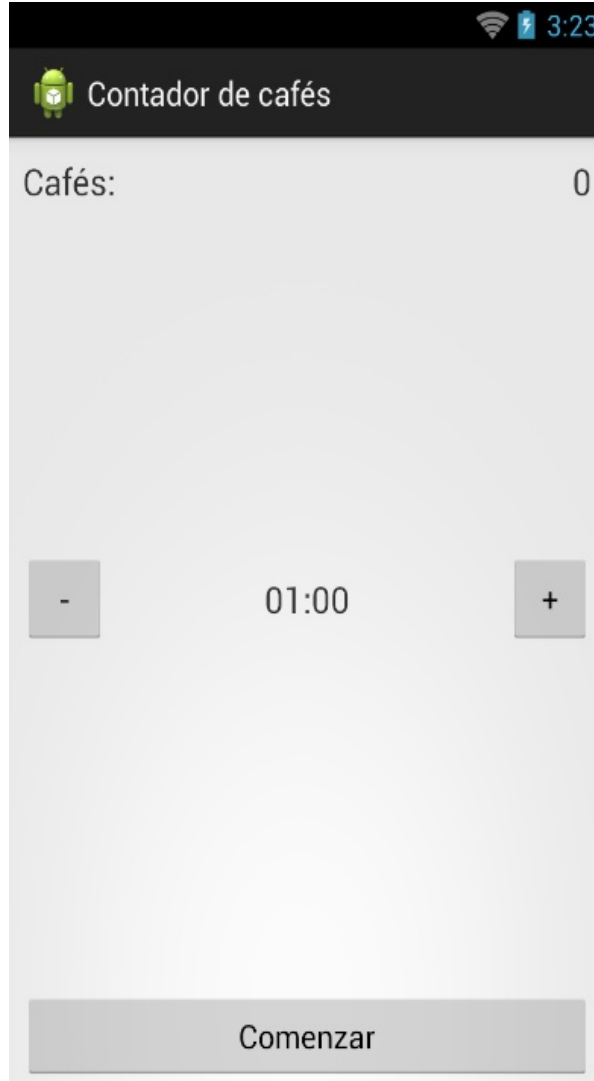
Contador de cafés

Crear una aplicación que muestre un temporizador para controlar la duración de la pausa para tomar un café.

Habrà un botón para iniciar el temporizador, se podrá configurar la duración de la pausa (en minutos) y se avisará al terminar el tiempo.

Además, contará los cafés tomados diariamente: se incrementará el contador al terminar cada pausa.

Contador de cafés



Fin

¿Dudas?

¿Sugerencias?



paco@portadaalta.es